linux内核编译及添加系统调用

系统调用表

- 用于关联系统调用号及其相对应的服务历程入口地址。系统调用-read

系统调用号	32/64/common	系统调用名称	服务例程入口
0	common	read	sys_read

- `path: /arch/x86/entry/syscalls/syscall_64.tbl(32位系统是syscall_32.tbl)`

一、下载新内核并编译、更换：

第一步：下载解压,进入文件夹

```
#wget <https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-#4.16.1.tar.xz>
#xz -d linux-4.16.1.tar.xz
#tar -xvf linux-4.16.1.tar
#cd linux-4.16.1
```

第二步： 清楚残留的 .config和 .o文件（每次编译出错或者重新编译最好都清理，不清理很占内存）

```
#make mrproper
```

报错提醒安装ncurses，重新执行make mrproper

```
#apt-get install libncurses5-dev
```

第三步：配置内核

```
#make menuconfig
```

根据报错提示安装组件，缺啥装啥

```
#sudo apt install build-essential //安装make和gcc等
#apt-get install libncurses5-dev     //安装ncurses-devel
#sudo apt-get install flex       //安装flex
#sudo apt-get install bison          //安装bison
```

没有报错后再执行

```
#make menuconfig
```

出现配置的对话框，直接保存（save），文件名也默认.config, 退出。

第四步：编译内核，生成启动映像文件

```
#make -j4 //-j4是用于加快编译速度。这里我是4核
```

报错提示要openssl，安装完再次执行命令即可

```
#apt-get install libssl-dev
```

第五步：编译模块

这一步要好久（2-3小时，可能虚拟机配置太低吧）。。。。睡一觉回来就好了

```
#make modules
```

第六步：安装内核、模块

```
安装模块：# make modules_install
安装内核：#make install
```

第七步：配置 grub 引导程序

只需要执行如下命令：该命令会自动修改 grub

```
#update-grub2
```

最后一步重启：

```

```

查看内核版本

```
#uname -a
```

```
lyq@lyq-virtual-machine:~/Documents/os$ uname -a
Linux lyq-virtual-machine 4.16.1 #1 SMP Sun Feb 23 23:28:45 CST 2020 x86_64 x86_
64 x86_64 GNU/Linux
lyq@lyq-virtual-machine:~/Documents/os$
```

成功更换内核!

二、添加简单系统调用

第一步：修改源程序

```
#cd linux-4.16.1 //进入linux解压包（我下的版本是4.16.1）
#vim arch/x86/entry/syscalls/syscall_64.tbl //进入该文件分配系统调用号 （注意别写在最后面，300多行，别写到下半部分的x32那一块里面）
```

```
327     64      preadv2                 sys_preadv2
328     64      pwritev2                sys_pwritev2
329     common  pkey_mprotect           sys_pkey_mprotect
330     common  pkey_alloc              sys_pkey_alloc
331     common  pkey_free               sys_pkey_free
332     common  statx                   sys_statx
333     64      mysyscall               sys_mysyscall
#
```

```
#vim include/linux/syscalls.h 进入该文件，添加服务例程的原型声明（shift+g快速跳到最后一行）
```

```
asmlinkage long sys_mysyscall(void);
#endif
```

```
vim kernel/sys.c 实现系统调用服务例程
```

```
SYSCALL_DEFINE0(mysyscall)
{
        printk("Hello, this is lyq's syscall test!\n");

        return 0;
}
#endif /* CONFIG_COMPAT */
```

第二步：编译安装内核

```
1. #make menuconfig 配置内核
2. #make -j2 编译内核
3. #make modules 编译模块
4. #make modules_install 和 make install 安装模块和安装内核
5. #update-grub2（好像虚拟机不需要这一步）
6. #reboot -n 立即重启
```

第三步：新系统调用测试

这里编写一个test.c文件来测试（文件存放位置可以任意）

```
#vim test.c
```

```
#define _GNU_SOURCE
#include <unistd.h>
#include <sys/syscall.h>
int main(){
        syscall(333);
}
~
~
~
```

编译

```
gcc test.c -o test //-o test指定编译输出文件名为test
```

```
lyq@lyq-virtual-machine:~/Documents/os$ ls
linux-4.16.1  linux-4.16.1.tar  test  test.c
```

执行文件

```
./test
```

查看信息

```
dmesg
```

```
3
64190.007894] usb 2-2.1: Product: Virtual Bluetooth Adapter
64190.007896] usb 2-2.1: Manufacturer: VMware
64190.007897] usb 2-2.1: SerialNumber: 000650268378
74498.857547] Hello, this is lyq's syscall test!
yq@lyq-virtual-machine:~/Documents/os$
```

可见系统调用成功执行

三：添加API对指定进程的 nice 值的读取功能

> 注：nice值表示进程可被执行的优先级的修正数值，加入nice值后，将会使得PRI变为：PRI(new)=PRI(old)+nice。这样，当nice值为负值的时候，那么该程序将会优先级值将变小，即其优先级会变高，则其越快被执行。

修改源程序

```
333     64      mysyscall                sys_mysyscall
334     64      mysetnice               sys_mysetnice
#
# x32-specific system call numbers start at 512 to avoid cache impa
```

```
asmlinkage long sys_mysyscall(void);
asmlinkage long sys_mysetnice(pid_t pid,int flag,int nicevalue,void __user * pri
o,void __user * nice);
#endif
```

pid：进程ID

flag：等于1表示修改，等于0表示读取

nicevalue：为指定进程设置新的nice值

prio，nice：指向进程当前优先级prio及nice值

```
SYSCALL_DEFINE5(mysetnice, pid_t, pid, int, flag, int, nicevalue, void __user *, prio, void _
_user *, nice)
{
        struct pid * kpid;
        struct task_struct * task;
        int nicebef;
        int priobef;
        kpid = find_get_pid(pid);  //return pid
        task = pid_task(kpid, PIDTYPE_PID);  //return task_struct
        nicebef = task_nice(task);  //get value of nice before change
        priobef = task_prio(task);  //get value of prio before change

        if(flag == 1){
                set_user_nice(task, nicevalue);
                printk("the value of nice before change: %d\tafter change: %d\n", nicebef, ni
cevalue);
                return 0;
        }
        else if(flag == 0){
                copy_to_user(nice, (const void*)&nicebef, sizeof(nicebef));
                copy_to_user(prio, (const void*)&priobef, sizeof(priobef));
                printk("The nice value of the process: %d\n", nicebef);
                printk("THe prio value of the peocess: %d\n", priobef);
                return 0;
        }
        printk("Invalid flag, which should be 0 or 1");
        return EFAULT;

}
#endif /* CONFIG_COMPAT */
```

copy_to_user函数则是从内核空间拷贝内容到用户空间，用户空间的进程无法直接访问内核空间的内容。这个函数做了数据合法判断。然后进行拷贝。

```
  static inline int task_nice(const struct task_struct *p)
```

用于获取当前task的nice值，并返回nice值，nice值的范围是[-20 ... 0 ... 19] 其使用的例程如下：

```
  void set_user_nice(struct task_struct *p, long nice)
  {
  bool queued, running;
  int old_prio, delta;
  struct rq_flags rf;
  struct rq *rq;

  if (task_nice(p) == nice || nice < MIN_NICE || nice > MAX_NICE)
  return;
  }
```

return -EFAULT代表返回一个错误代码；

pid_t 其实就是__pid_t类型。

```
  #ifndef __pid_t_defined
  typedef __pid_t pid_t;

  define __pid_t_defined

  #endif
```

编译安装内核

。。。

编写测试程序：

lyq@lyq-virtual-machine: ~/Documents/os

```
#define _GNU_SOURCE
#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>

int main()
{
        pid_t tid;
        int nicevalue;
        int prio = 0;
        int nice = 0;
        tid = getpid();
        syscall(334, tid, 0, -5, &prio, &nice);//read
        printf("read(-5):pid:%d prio:%d nice:%d\n",tid, prio, nice);
        syscall(334, tid, 1, -5, &prio, &nice); //set
        printf("read(-5):pid:%d prio:%d nice:%d\n",tid, prio, nice);
        syscall(334, tid, 0, -5, &prio, &nice);//read
        printf("read(-5):pid:%d prio:%d nice:%d\n",tid, prio, nice);

        return 0;
}
~
~
"nice-test.c" 21L, 514C                                       1,1           All
```

```
lyq@lyq-virtual-machine:~/Documents/os$ ./nice-test
read(-5):pid:3089 prio:20 nice:0
read(-5):pid:3089 prio:20 nice:0
read(-5):pid:3089 prio:15 nice:-5
```

```
[   52.666598] The nice value of the process: 0
[   52.666599] THe prio value of the peocess: 20
[   52.666659] the value of nice before change: 0       after change: -5
[   52.666663] The nice value of the process: -5
[   52.666663] THe prio value of the peocess: 15
lyq@lyq-virtual-machine:~/Documents/os$
```